frivol

Generated by Doxygen 1.8.1.2

Thu May 16 2013 12:28:44

Contents

Chapter 1

Todo List

Member frivol::fortune_::Algorithm< PolicyT >::finish ()

Currently unfinished, therefore doesn't return anything.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 frivol::fortune_ Namespace Reference

Classes for the implementation of Fortune's algorithm.

Classes

- struct Arc

    Information of an arc in the beach line.
- class Algorithm

Chapter 5

Class Documentation

5.1 frivol::fortune_::Algorithm< PolicyT > Class Template Reference

`#include <frivol_impl.hpp>`

Public Types

- typedef PolicyT::Coord **CoordT**
- typedef Point< CoordT > **PointT**

Public Member Functions

- Algorithm (const Array< PointT > &sites)
- void step ()

  Run the algorithm one event handling forward.
- CoordT getY ()
- bool isFinished ()

  Returns true if the algorithm has finished.
- void finish ()

5.1.1 Detailed Description

template<typename PolicyT>class frivol::fortune_::Algorithm< PolicyT >

State of Fortune's algorithm.

Template Parameters

PolicyT	The algorithm policy to use, instance of Policy template.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 template<typename PolicyT > frivol::fortune_::Algorithm< PolicyT >::Algorithm (const Array< PointT > & *sites*) `[inline]`

Constructs algorithm state.

Parameters

points	Reference to the input set of sites. The object must exist throughout the existence of the Algorithm.

5.1.3 Member Function Documentation

5.1.3.1 template< typename PolicyT > void frivol::fortune_::Algorithm< PolicyT >::finish ( ) `[inline]`

Steps the algorithm until the end and returns the result.

Todo Currently unfinished, therefore doesn't return anything.

5.1.3.2 template< typename PolicyT > CoordT frivol::fortune_::Algorithm< PolicyT >::getY ( ) `[inline]`

Get the y coordinate of last step(). Undefined return value if step() has not been called yet.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/frivol_impl.hpp

5.2 frivol::fortune_::Arc Struct Reference

Information of an arc in the beach line.

`#include <frivol_impl.hpp>`

Public Attributes

- Idx site

  The site from which the arc originates.
- Idx id

  The ID of the arc.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/frivol_impl.hpp

5.3 frivol::Array< T > Class Template Reference

`#include <array.hpp>`

Public Member Functions

- Array (Idx size)
- Idx getSize () const

  Returns the size of the array.
- void resize (Idx size)

- const T & operator[] (Idx index) const
- T & operator[] (Idx index)

5.3.1 Detailed Description

template<typename T>class frivol::Array< T >

Simple fixed-size array.

Template Parameters

T	The type of stored elements. Should be default constructible.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 template<typename T > frivol::Array< T >::Array (Idx *size*)

Creates an array with all elements default-constructed.

Parameters

size	The size of the array.

5.3.3 Member Function Documentation

5.3.3.1 template<typename T > const T & frivol::Array< T >::operator[] (Idx *index*) const

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

5.3.3.2 template<typename T > T & frivol::Array< T >::operator[] (Idx *index*)

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

5.3.3.3 template<typename T > void frivol::Array< T >::resize (Idx *size*)

Resizes the array to size. If size decreases the extra elements are removed. If size increases, the new elements are default-constructed. The operation may assign the current elements to a new place, and therefore pointers to the array may be invalidated.

Parameters

size	The new size.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/array.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/array_impl.hpp

5.4 frivol::DummyPriorityQueue< PriorityT > Class Template Reference

Simple implementation of PriorityQueueConcept.

```
#include <priority_queue_concept.hpp>
```

Public Member Functions

- **DummyPriorityQueue** (Idx size)
- std::pair< Idx, PriorityT > **pop** ()
- bool **empty** () const
- void **setPriority** (Idx key, PriorityT priority)
- void **setPriorityNIL** (Idx key)

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/priority_queue_concept.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/priority_queue_concept_impl.hpp

5.5 frivol::DummySearchTree< ElementT > Class Template Reference

Simple implementation of SearchTreeConcept (a wrapper around std::list).

```
#include <search_tree_concept.hpp>
```

Public Types

- typedef std::list< ElementT >
  ::iterator **Iterator**

Public Member Functions

- template<typename FuncT >
  Iterator **search** (FuncT func)

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/search_tree_concept.hpp

5.6 frivol::Point< CoordT > Struct Template Reference

```
#include <point.hpp>
```

Public Member Functions

- Point (CoordT x, CoordT y)
- Point ()

  Constructs point with undefined values as coordinates.

Public Attributes

- CoordT x

  The X coordinate of the point.
- CoordT y

  The Y coordinate of the point.

5.6.1 Detailed Description

template<typename CoordT = DefaultPolicy::Coord>struct frivol::Point< CoordT >

Two-dimensional point.

Template Parameters

CoordT	The coordinate type to use. Should be default constructible. Defaults to the coord type of the default policy.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 template<typename CoordT = DefaultPolicy::Coord> frivol::Point< CoordT >::Point (CoordT *x*, CoordT *y*) [inline]

Constructs point with given coordinates.

Parameters

x	The X coordinate.
y	The Y coordinate.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/point.hpp

5.7 frivol::Policy< CoordT, EventQueueT, BeachLineT > Struct Template Reference

```
#include <policy.hpp>
```

Public Types

- typedef CoordT **Coord**

5.7.1 Detailed Description

template<typename CoordT, template< typename PriorityT > class EventQueueT, template< typename ElementT > class BeachLineT>struct frivol::Policy< CoordT, EventQueueT, BeachLineT >

Policy class for the Fortune's algorithm, specifying data types and data structures to use.

Template Parameters

CoordT	The coordinate type to use. Should be ordered and default constructible to undefined value. Should have specialization of GeometryTraits.
EventQueueT	The priority queue type for events. Must conform to PriorityQueueConcept.
BeachLineT	The search tree to use for the "beach line" of arcs. Must conform to SearchTreeConcept.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/policy.hpp

5.8 frivol::PriorityQueueConcept< X, PriorityT > Class Template Reference

```
#include <priority_queue_concept.hpp>
```

5.8.1 Detailed Description

template<typename X, typename PriorityT>class frivol::PriorityQueueConcept< X, PriorityT >

Concept checking class for priority queues X with priority values of type PriorityT (or NIL). Priority queues are initialized with given size, and contain priority values for keys 0, 1, ..., size-1. Initially, all priority values are NIL. X must support the following operations:

- <construct>(Idx size) creates priority queue for keys 0, 1, ..., size-1.
- bool empty() const returns true if all keys have NIL priority.
- std::pair<Idx, PriorityT> pop() returns pair of a key with lowest non-NIL priority and its priority and sets the priority to NIL.
- void setPriority(Idx key, PriorityT priority) sets the priority value of 'key' to non-NIL value 'priority'.
- void setPriorityNIL(Idx key) sets the priority value of key 'key' to NIL.

X may assume that PriorityT is ordered with <-operator. X may have undefined behavior if supplied keys are out of range or if pop() is called when empty() returns true.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/priority_queue_concept.hpp

5.9 frivol::SearchTreeConcept< X, ElementT > Class Template Reference

```
#include <search_tree_concept.hpp>
```

Public Types

- typedef X::Iterator **IteratorT**

5.9.1 Detailed Description

template<typename X, typename ElementT>class frivol::SearchTreeConcept< X, ElementT >

Concept checking class for search trees X for elements of type ElementT. Search trees are sequence containers, the elements of which are iterated using iterator objects of type X::Iterator. The iterator must be a standard bidirectional iterator. X must support the following operations:

- <construct>() creates empty search tree.
- bool empty() const retuns true if the search tree is empty.
- Iterator begin() returns the iterator of the first element (or past-the-end if empty).
- Iterator end() returns the iterator past the last element.
- template<typename FuncT> Iterator search(FuncT func) searches the sequence using the supplied int(-Iterator)-function that for given iterator iter returns negative if the searched element is before iter, positive if it is after iter, and 0 if iter is the right element. If an element such that func returns 0 is found, it is returned, otherwise end() is returned.
- void erase(Iterator iter) removes element at iter. Other iterators should not be invalidated.
- void insert(Iterator iter, const ElementT& elem) inserts elem before iter. Does not invalidate any iterators.

X may assume that ElementT is copy constructible.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/search_tree_concept.hpp

5.10 frivol::Stack< T > Class Template Reference

```
#include <stack.hpp>
```

Public Member Functions

- Stack ()

    Constructs empty stack.
- bool empty () const

    Returns true if the stack is empty.
- T & top ()
- void pop ()

    Removes the top element of the stack. Call only if empty() is false.
- void push (const T &element)

5.10.1 Detailed Description

template<typename T>class frivol::Stack< T >

Stack of elements.

Template Parameters

T	The type of stored elements. Should be default constructible.

5.10.2 Member Function Documentation

5.10.2.1 template<typename T> void frivol::Stack< T >::push (const T & *element*)

Pushes element to the top of the stack.

Parameters

element	The element to push.

5.10.2.2 template<typename T > T & frivol::Stack< T >::top ( )

Returns reference to the top element of the stack. Call only if empty() is false.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/stack.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/stack_impl.hpp